# Automated Planning for Military Airline Controller Training Scenarios

Romain Goutiere, Domitile Lourdeaux[a] and Sylvain Lagrue[b]
*Alliance Sorbonne Université, Université de Technologie de Compiègne, CNRS, Heudiasyc UMR 7253, CS 60 319, 60203 Compiègne Cedex, France*

Keywords: ANML, Automated Scenario Planning, Interactive Storytelling, Alternative Scenario Generation, HTN.

Abstract: In this paper we focus on the generation of scenarios for military airline controller training in a virtual environment. We are using a planning system mixing temporal planning and hierarchical task networks based on the ANML planning language and allowing a better representation of both narrative and pedagogical objectives. We also propose and test a method to automatically generate potential alternative plans to the initially planned scenario, reaching the same objectives, to make it more robust.

## 1 INTRODUCTION

The objective of **Interactive Storytelling (IS)** is to create multimedia systems in which users can interact and influence, in interactive time (time interval between users actions), the evolution of the narrative. One of the most active research areas in IS is the **automated generation of scenarios**. The objective of generative approaches is to reduce the authoring bottleneck (difficulties for an author to design complex scenarios with multiple storylines and interactions) during the design of the interactive narration system. The scenario is thus automatically generated before and/or dynamically during the simulation, opening the way to adaptations of the scenario during the simulation and to a greater variety in the proposed content. **Planning** is nowadays a widespread method for automatic scenario generation. However, its use in IS requires some adaptations compared to more classical application frameworks and several challenges remain. Porteous (Porteous, 2016) highlights in particular: (1) the planner's ability to respond to narrative content, (2) interactive time control over plan generation, (3) user interactivity which requires the system to be resilient, and (4) planning on qualitative criteria other than pure optimization. Two planning methods are mainly used in IS. **Classical planning**, based on the use of **heuristics** and bringing a strong variability due to the generative power of this method. And **hierarchical task networks (HTN)**, consisting in a successive decomposition of the problem and bringing a high degree of control on the generated plans. Each of these methods has useful characteristics in IS, but to date there is no framework allowing the use of both systems at the same time.

IS systems are also useful for training and education. Gupta (Gupta et al., 2008) gives several advantages such as being able to more easily put learners in a variety of situations, and to have them repeat as needed. However, systems providing training must meet certain criteria to ensure the effectiveness of the experience. For example, it is important that the difficulty is well calculated, that the scenarios are **coherent** and **explainable** for the trainer, etc.

In this paper, we detail our approach for the automatic planning of training scenarios within the framework of this project with the use of the **ANML** planning language allowing a **hybrid planning**, very adapted to combine **generative power** and **scenario control**. We also propose algorithms allowing the generation of **offline alternatives**, bringing **robustness** to the proposed training scenarios. We will first describe the ORCHESTRAA project, to which our work is linked. Then we will explain the different approaches used for scenario planning as well as the problems related to the design of scenarios and their robustness in an interactive environment. Then, we will detail our different contributions.

## 2 THE ORCHESTRAA PROJECT

Our work is part of the **ORCHESTRAA** project (Orchestration of stessful situations for training based on

[a] https://orcid.org/0000-0002-3354-7294
[b] https://orcid.org/0000-0001-9292-3213



Goutiere, R., Lourdeaux, D. and Lagrue, S.
Automated Planning for Military Airline Controller Training Scenarios.
DOI: 10.5220/0011688700003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 466-474
ISBN: 978-989-758-623-1; ISSN: 2184-433X

virtual reality and adaptative agents in air operations context) which aims at developing an interactive virtual reality environment reproducing an air operations control center. This environment must allow users (immersed in the environment through a HMD) to carry out training sessions that are characterized by the management of different conflict situations, involving air and ground troops on a fictitious theater of operations. This theater of operation is managed by an external simulator interacting with the operators through visualization tools available on the terminals of the different stations and through a chat.

Our work within this project is to design a **scenario system** allowing: the **automatic generation of training scenarios** executed both in the virtual environment and the simulator, the control of the scenario execution and the live monitoring of the scenario by the trainers.

# 3 STATE OF THE ART

## 3.1 Scenario Planning

Planning methods for scenario generation are widely used today. It has several advantages: **causality**, **generative power** and **narrative structure** (Porteous, 2016). In the context of IS, causality refers to the relation between temporal and ordered events that allow to predict the occurrence of future events. Generative power represents the ability of a system to generate varied content. Finally, the narrative structure allows to define the key stages in the progression of a story.

Classical planning is widely used in IS. A plan is then a succession of actions and events in order to reach a desired goal. Classical planning problems are solved with **Heuristic Search Planner (HSP)** following a heuristic to find the best plan. There are then many variations aiming at handling more complex cases such as the addition of temporal constraints (Porteous et al., 2011) allowing to add duration, or to parallelize actions. Another interesting method is also used in IS (Aylett et al., 2006): **hierarchical task networks (HTN)**. The plan of an HTN problem is not represented by a sequence of events and actions, but as a successive decomposition of complex actions into simple actions. The structure of a plan is then presented as a tree.

These two planning methods each offer interesting features in interactive narration, but which may be problematic depending on the context. Classical planning, using heuristics, offers great generative power. Indeed, the planner will work by making a selection on the set of available actions, which allows for a high variability, good replanning capabilities and a better resilience. However, this generative power also forces a difficult reflection work during the design phase in order to avoid the **generation of illogical plans** from a narrative point of view. This constraint is the more important in a training context where it is essential to propose scenarios having a pedagogical interest.

On the other hand, HTNs emphasize the respect of the **author's intention**. Indeed, a HTN domain is made of a set of successive decompositions. The planner chooses the actions among the proposed decompositions and not among all the actions of the domain. HTNs are also quite simple to model. It is sufficient that the problem is decomposable. For each decomposition the alternatives are explicitly defined. Thus, whatever the choice of the planner, the final plan is made of elements foreseen by the author. These features are of interest for IS and seem to be adapted to the context of training and coaching. However, the generative power of this method is less efficient than classical planning. Each of the alternatives has to be imagined by the author, a **bottleneck authoring** problem quickly arises.

Table 1 summarizes the advantages of these planning methods in relation to interesting features in IS.

Table 1: Comparison HTN/HSP.

| | HTN | HSP |
|---|---|---|
| Author intention | + | - |
| Resilience | - | + |
| Variability | - | + |
| Replanification | - | + |
| Modelling | + | - |

## 3.2 Planning Languages

To solve an IS problem by a planning method, it is necessary to express it in a language that a planner can understand. In order to be able to compare the performance of different planners, standardization projects of these languages have been proposed. The current standard planning language is the Planning Domain Description Language (PDDL) (Howe et al., 1998), which is currently used in International Planning Competitions (IPC). However, the domains/problems studied in IS are often **more complex** than those studied in classical planning because they describe concepts of various kinds, which generally translate into a larger number of actions, difficulties in representing the state of the world without **losing too much information**, or the addition of many constraints. This difference imposes certain choices during the formalization of the domain/problem which can lead to an overall decrease in terms of **intelligibility** for the human user, and a weakening in the **ex-**

**pressiveness** of the scenario.

The main interest of PDDL is to propose a **standard** that can be understood by the largest possible number of planners. However, PDDL has **limitations**, such as the impossibility of simply representing hierarchical planning problems or difficulties in representing complex states of the world containing concepts of very different natures. Moreover, the **quality criteria** of plans are not the same in IS and in classical planning, which implies that the design of PDDL is oriented towards performances rather than other criteria, more related to narration and often specific to the studied domains.

The **Action Notation Modeling Language (ANML)** (Smith et al., 2008) is an interesting option to bring better modelling to IS problems. It is a planning language offering higher level features than PDDL. Specifically, it allows the use of quite complex temporal constraints and also allows modeling and solving planning problems represented in both **classical and hierarchical** forms. It is thus possible to develop **hybrid planning** domains where it is not necessary to adapt the modeling to one or the other system, but where each element is represented according to the approach that best corresponds to it. This brings great **expressiveness** in a field such as IS, where rather different concepts must run together.

### 3.3 Robustness of the Scenario

In an IS system, the running of the scenario can be problematic. Indeed, a scenario must rely on certain actions of various actors (human users, virtual characters). Thus, a scenario can potentially be blocked if an action to be performed by a user never happens.

Many strategies have been developed to address this problem. They are now gathered around the concept of **Experience Management** (Riedl and Bulitko, 2013) which consists of monitoring the running of the scenario and setting up intervention strategies to enable it to continue in the best possible way.

One of the main strategies, the **narrative mediation** (Riedl et al., 2003), proposes, as long as it is possible, to let the actors perform their actions freely. If these actions enter into conflict with the planned scenario, the scenario is **replanned** from the point of divergence to take into account the problematic actions, in order to solve the **same objectives** as the initial scenario.

However, this solution is not perfect, because it must be applied in interactive time during the simulation. This often implies the use of fast algorithms that do not necessarily guarantee the **quality** of the new plan (not enough time available to analyze the construction of the plan from a narrative point of view) (Ramirez and Bulitko, 2014).

Finally, another approach aims at **predicting** potential errors that could lead to scenario problems and generating offline directly applicable alternatives. Automated Story Director (Riedl et al., 2008) is a good example. In this system, a scenario is made up of events of low importance and of Island, which are events most important for the scenario. The principle is, for each action, to foresee situations that could be dangerous for the next Island and to generate an alternative to reach it.

## 4 OUR PROPOSALS

The issues raised by the ORCHESTRAA project are quite representative of the current challenges of using planning methods in IS and can be extended to other problems. We need the **generative power** brought by planning to generate varied scenarios but the training context also forces us to be careful about the emergence of potentially problematic content from a pedagogical point of view. The interactive aspect of the system also raises the question of the **robustness** of our scenarios and their **resilience** with respect to the pedagogical objectives of training. Moreover, the question of formalizing a scenario content into a planning language remains a task that requires a lot of work to express concepts that are sometimes not adapted to a given type of formalism.

Our proposals provide a direct response to these problems. First, we propose a new **representation of scenarios** more adapted to our needs thanks to the use of ANML. We detail the functioning of this new structure and its advantages for the representation of our scenarios.

We also present an approach for **automatically generating offline alternatives**, based directly on the structure of our scenarios to increase their robustness and resilience with respect to their **pedagogical objectives**.

### 4.1 Planning Methods and Hybrid Planning

Choosing a planning method is not easy. Classical planning offers great generative power. It is an interesting choice for our system because it allows to easily generate different training sessions. This choice also has the advantage of facilitating re-planning if user interactions cause a drift from the planned scenario. However, it is not easy to design a planning domain, reaching the needs of the project, adapted to

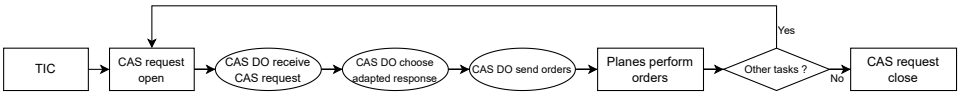


Figure 1: CAS process.

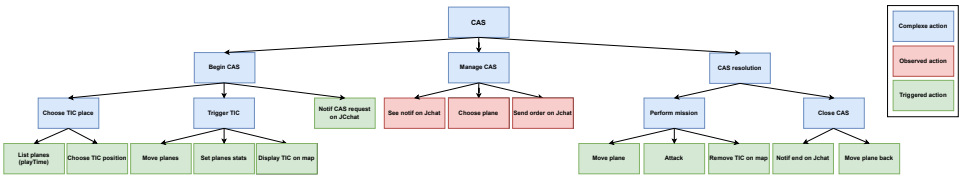


Figure 2: CAS representation as HTN.

this method. Indeed, the training situations proposed to military air traffic controllers are decomposed according to a rather strict order. Figure 1 gives a protocol for the processing of a Close Air Support (CAS) request, a frequent element of our training scenarios.

All the actions in the planning domain would be constrained by a fixed order, and would not benefit from the generative power of planning. These constraints would be all the more important as the slightest oversight or imprecision could lead to unforeseen **emergent situations**.

The protocol presented in Figure 1 is high level. In order for it to be instantiated in the virtual environment, it is necessary to decompose it into lower level actions. The HTN representation therefore seems to be a better solution. Indeed, HTNs are particularly well adapted to frozen processes such as those we are interested in. Figure 2 shows the beginning of the HTN decomposition of a CAS.

These CASs do not constitute a complete training scenario. A complete scenario is a set of CASs whose treatment varies slightly from one to another in order to work on a set of desired skills. These variations are easily represented in HTN by explaining alternative decompositions for certain actions. However, a scenario containing a succession of different CAS is difficult to represent in HTN. Indeed, an HTN problem is the decomposition of a single complex task into simple tasks. In order to represent our sessions, we would have to decompose a complex ”session” action into several ”CAS” actions, each one linked to a particular skill. This representation poses a major problem in case of need for replanning because it requires to make explicit all the different possibilities of arrangement of the session at the level of skill, which poses a problem of expressiveness and does not correspond to the logic of HTN.

On the contrary, classical planning allows a simpler representation of these skills and the generative power provided allows to generate much more simply all the alternatives in terms of training session layout.

We thus have two planning methods, each with advantages in some aspects and deficiencies in others. The usual approach to solve this kind of problem would be to make a choice and adapt our representation to **minimize the deficiencies** of the representation. This method, although widely used, is not good. We therefore propose another way, made possible by the Action Notation Modelling Language (ANML), to represent **hybrid classical/HTN** planning problems. Thanks to this language, we can handle the pedagogical contents via a classical planning method and the scenaristic aspects with HTN decompositions. The link between the two methods is made through a planning operator that we have developed: **Learning Unit**.

## 4.2 New Operator : Learning Unit

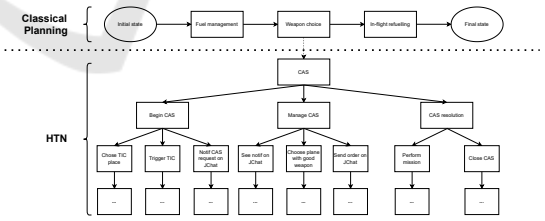


Figure 3: Hybrid scenario structure.

A **Learning Unit (LU)** is a high-level planning operator used to link classical planning to a hierarchical planning problem. A LU is characterized by one or more **target skills**, a **start time**, a **duration** and an associated **hierarchical decomposition**.

The ANML allows us to define a duration constraint expressed as an interval. Two preconditions are also necessary. The first one is simply to check that the training will focus on the desired skill. The second is more interesting:

```
[all] isValidated(v1) ==
    false :-> true;
```

This constraint indicates in a simple way that the skill must not have been validated before the LU and must be validated at an unspecified time during the LU in order for this validation to be verified at the end of the LU.

Finally, the last part indicates that the action is also the root of a hierarchical decomposition. This decomposition is only done in one element, the CAS, which will in turn be decomposed.

This structure is not usual in planning. In normal time, the root of a HTN problem has no conditions, its resolution coming only from its decomposition. In the same way, a classical planning problem does not decompose in a hierarchical way. This structure allows us to propose a **strict separation** between the scenaristic and pedagogical content, and thus allows us to use the planning method most adapted to each content.

## 4.3 Global Plan

With our scenario structure, a plan can be summarized by a succession of LUs, each of which is decomposed into an HTN. Figure 3 shows us the global structure of a scenario.

For the moment, we have detailed the advantages of such a structure for the representation of different types of narrative content. However, the advantages of this structure are more extensive. First of all, this structure brings a better expressiveness to the scenario. Indeed, it is possible to give precisely the state of the progression from a pedagogical and scenaristic point of view. This information is precious for the trainers who follow the progress of the training sessions. Thanks to this structure, we can provide them with this information in a simple way.

This structure also has advantages concerning the robustness of the scenario. Indeed, this structure allows to set up **replanning strategies** adapted to each type of error and not systematically requiring a total replanning of the plan. At the HTN level, for example, the alternatives to each decomposition are made explicit. If the state of the world allows them to be planned, they constitute alternatives that can be used directly without having to replan the entire scenario. Such alternatives are simply represented in ANML :

```
action sendOrder(Learner l1,
    Order o1) {
  duration :in [3, 8];
  [all] orderSent(o1) ==
    false :-> true;
  :decomposition {
    [start + 2] learnOnChat(l1) ==
        true;
    writeOrder(l1, o1),
    sendToChat(o1)
  };
  :decomposition {
    [start + 2] learnOnChat(l1) ==
        false;
    helpToLearner(l1),
    writeOrder(l1, o1),
    sendToChat(o1)
  };
};
```

In this example we see two different decompositions for the `sendOrder` action which corresponds to the sending of an order by the learner on the chat window. In the first decomposition, the learner is on the chat window after 2 units of time while he is not in the second one. The second decomposition then adds a helper action allowing the learner to perform the right actions. If the state of the world allows it, these two decompositions thus allow to plan two different scenarios. However, these two scenarios do not challenge the higher level elements of the HTN, nor the organization of the LUs.

Thus, as long as there are alternative **plannable decompositions**, replanning is limited to the HTN problem and does not affect the rest of the scenario.

However, if replanning is not possible at the HTN level (no plannable decompositions with the state of the world), the HTN can be considered to have failed because the decomposition can no longer be fully realized. Such a problem leads to a failure of the post-conditions of the LU (which corresponds to the root of the HTN). From a pedagogical point of view, this situation corresponds to the fact that the skill provided by this LU has not been validated. It is then necessary to have a set of **alternative plans** taking into account the failures on the different LUs.

## 4.4 Automatic Generation of Alternatives

Our objective is to generate **offline** a set of alternative plans to the initial plan so that in case of drift during the execution of the scenario, the system can switch to another plan already generated, adapted to the current situation and satisfying the same objectives as the initial scenario. We have already seen that the generation of alternatives at the HTN level is quite trivial. Either there are other plannable decompositions and in this case there is an alternative plan (the system can

Algorithm 1: Failure tree generation.

```
Data: LU_list_err, d_max, err_max
Result: error_tree
struct (
    int: d
    LU: pre, potential_err[], next[]
) LU
LU : root
root.d ← 0
foreach LU ∈ LU_list_err do
    root.d ← root.d + LU.d
end
root.pre ← NULL
root.potential_err ← LU_list_err
current_err ← root
while current_err ≠ NULL do
    if next_LU ←
     pop(current_err.potential_err) then
        UA : next_err
        next_err.d ← next_LU.d
         + current_err.d
        if next_err.d <= d_max &
         LU_list_err[next_LU][0] + 1 <=
         err_max then
            LU_list_err[next_LU][0] += 1
            current_err.next.add(next_err)
            next_err.pre ← current_err
            current_err ← next_err
        else
            current_err ← current_err.pre
        end
    else
        current_err ← current_err.pre
    end
end
return root
```

choose the best adapted alternative decomposition) or there is not, and therefore the LU during which the drift occurs is considered as failed (associated skill not worked).

It is therefore necessary to take this failure into account and to have alternative plans at the LU level. However, the problem is not as trivial as for HTN. Indeed, two elements are likely to generate a combinatorial explosion problem when we want to generate all possible alternatives. Firstly, without additional constraints, it is theoretically possible to have an indefinite number of failures on each LU. Second, since LUs are relatively unconstrained actions, there is a large number of potentially plannable permutations.

We must therefore implement a strategy to reduce the set of possibilities. Regarding the undefined number of failures, we can consider two constraints. First, a plan represents a training session. This session is therefore limited in time. Knowing that each LU is also characterized by a duration, as soon as we define a general time limit for the training session, it can only contain a limited number of LUs (success or failure). The second constraint consists in adding a maximum error limit per LU. It is a heavier constraint but it is logical in our application framework. Thanks to these two constraints, it is possible to generate a tree containing all the LU sets (success and failure).

Algorithm 1 shows how to generate such a tree. Its operation is quite simple. As input we need an array containing the list of LUs with an integer associated to each representing the number of failures, the maximum duration of the scenario and the maximum number of errors per LU. To start we initialize the root which represents the successful LUs. Then for each possible failed LU we add the duration and increment the number of errors on this LU in the table. If the maximum values are not exceeded we add a new node which becomes the current node. If the constraints are not respected for all the potential failed LUs we go back and explore other branches. The tree is thus generated in depth first. The tree thus generated gives us all the possible error sets and all the arrangements. It is then necessary to remove the duplicates to obtain only the unordered failure sets.

The sets generated by this way contain the successful and failed LUs but do not order them. It is therefore necessary to generate the possible permutations of LUs for each of these sets. Once the set of permutations is generated it is then necessary to modify our planning domain so that the planner can determine the plannable permutations.

To simplify the work of the planner we perform a preprocessing of the permutations to remove those that do not meet the most trivial conditions. Two types of permutations are thus eliminated: first, permutations whose first LU is different from the initial plan (success or failure). Alternative plans are used in case of scenario drift, which can only occur once the execution of the plan has started, so the first LU has no reason to be different. Then, we can simply delete the permutations that place one or more successful LUs before these same failed LUs. Indeed it would be illogical to propose again a LU that has already been successfully processed.

This strategy allows to substantially reduce the number of permutations that will actually be processed by the planner. The generated plans constitute a set of possible organizations for the training session. The connection with HTNs and the available alternatives thus offer us the advantages brought

by the generative power of classical planning and the control brought by HTNs without being subject to the drawbacks of these two methods.

# 5 ALTERNATIVE GENERATION EXAMPLE

With our method we are able to generate a large number of potential alternative plans. We would like to test our system with a simple planning domain to see the generative power of our system and its limits.

## 5.1 Domain and Parameters

For this test we use a simple planning domain, only composed of Learning Units characterized by a duration and linked to a skill.

With this domain, we generated a plan on which we can rely to generate potential alternatives. The purpose of this test is to observe the number of alternative plans generated as a result of the number of Learning Units in the plan, the maximum duration of the training session and the maximum number of errors envisaged for each LU.

We thus wish to show that, given the particular constraints related to the planning of training scenarios, it is possible to use a **combinatorial** method to generate **offline** all the possible alternatives to the initial scenario, with a given configuration of the constraints.

Indeed, each Learning Unit is characterized by a duration. This duration concretely represents the time allocated to training on a given skill. A training session is therefore also characterized by a duration corresponding to the sum of the durations of the different Learning Units to which it is necessary to add a margin to manage potential errors. This time constraint is quite strong in the context of the generation of possible alternatives because when an error occurs that calls into question the success of a LU, it is necessary to reschedule it later in an alternative plan, and this implies a new plan consisting of an additional LU. Given the time limit for the session, the number of errors is therefore **limited**. In this context it is useful to also add a constraint on the maximum number of errors per LU, in order not to favor one skill over another (unless this is desired, in which case the maximum number of errors can be adapted for each LU). Finally, the initial planning domain also reflects these constraints. Thus, the smaller the margin between the session duration limit and the optimal duration of the initial plan, the less possibility there is to generate potential alternatives. On the contrary, the larger the margin and the larger the number of potential alternative plans, it is then interesting to show the limits of this approach, when the number of generated plans becomes too high for the system to calculate them in an **acceptable** time.

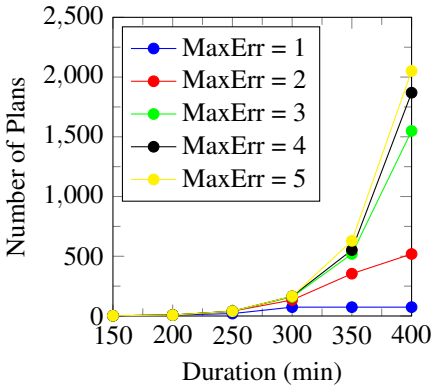


Figure 4: Offline generation with 3 LU.

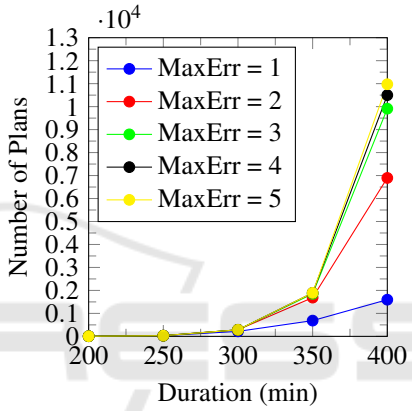


Figure 5: Offline generation with 4 LU.

We performed this test with duration constraints representative of our application context (military air traffic controllers training session). Thus, we generated 3 plans including 3 and 4 Learning Units. These Learning Units have respective duration ranging from 30 to 70 minutes for training sessions lasting between 100 and 400 minutes. For each plan we generated all the potential alternatives by varying the duration between the minimum time to perform the session (addition of the LU duration) and the maximum duration of the training session (400 minutes). We also varied the maximum number of errors allowed for each LU between 1 and 5.

### 5.2 Results and Analysis

We generated alternatives to the 3 initial plans by varying the duration with a step of 50 minutes. For each of these durations we varied the maximum number of errors for each LU. What can be observed first, for the 2 different cases, is that the number of generated plans remains quite low for $Duration < 300$ with a number of generated plans lower than 500 whatever the initial plan and the number of allowed errors. The number of generated alternatives increases sharply between 300 and 400 minutes, but the constraints still allow the plans to be generated without problems. We also note that with such duration constraints, the number of generated plans varies little for $MaxErr > 3$. Although the computations run smoothly for the duration interval $[100-400]$, there is a large variation in the number of generated plans depending on the length of the initial plan which varies from a few hundred for an initial plan composed of 3 LU to more than 10000 for an initial plan composed of 4 LU. These results therefore show that our method works for generating alternatives to the initial plan. The number of generated plans is high and offers many possibilities in order to select the most suitable alternative plan for the encountered situation. The maximum duration used for our tests is 400 minutes. However, this time is only reached when several errors are made and the initial plan, if it is carried out without errors, is much faster. It is therefore necessary to think carefully about the skills that the trainer wants to work on and the parameters of the limits according to the type of session that the trainer wants to set up. Thus, thanks to our system, it is quite possible to propose a rather short session but leaving the possibility of many attempts. This allows the learner to take more time, or, in case of quick success, to start a new training session. Alternatively, our system also allows for longer sessions, with more skills being worked on, but with less opportunity for mistakes to be made. These possibilities of adaptation make it possible to offer training sessions to different learner profiles such as beginners or experienced operators.

## 6 CONCLUSIONS

In this paper we have presented an original solution to design and plan **training scenarios**. **Hybrid planning** and the use of **ANML** allow us to take advantage of both the **generative power** of planning and the **control** provided by HTNs. This structure also greatly improves the **expressiveness** of our plans thanks to the separation of scenario and pedagogical content. Finally, this method improves the **robustness** of our scenario and its **resilience** with respect to its objectives thanks to an efficient generation of alternative plans allowing to anticipate the potential drifts of the scenario and to repair them. Finally, these elements were integrated into an interface allowing the trainer to follow the scenario execution live and to apply the generated alternatives in case of drift. In order to complete the system, we will study in our next work the strategies of qualitative selection of the generated alternatives so that the repair of the scenario corresponds as well as possible to the initial objectives and the preferences of the trainers. This work will be accompanied by the development of the monitoring system to support the execution of the scenario, to detect drift situations and to implement the alternatives.

## ACKNOWLEDGEMENTS

This work is financed by the DGA RAPID ORCHESTRAA project.

## REFERENCES

Aylett, R., Dias, J., and Paiva, A. (2006). An affectively driven planner for synthetic characters. In *Icaps*, pages 2–10.

Gupta, S. K., Anand, D. K., Brough, J. E., Schwartz, M., and Kavetsky, R. (2008). *Training in Virtual Environments: A Safe, Cost-Effective, and Engaging Approach to Training*. University of Maryland.

Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., SRI, D. W., Barrett, A., Christianson, D., et al. (1998). Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*

Porteous, J. (2016). Planning technologies for interactive storytelling. In *Handbook of Digital Games and Entertainment Technologies*. Springer.

Porteous, J., Teutenberg, J., Charles, F., and Cavazza, M. (2011). Controlling narrative time in interactive storytelling. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 449–456.

Ramirez, A. and Bulitko, V. (2014). Automated planning and player modeling for interactive storytelling. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):375–386.

Riedl, M., Saretto, C. J., and Young, R. M. (2003). Managing interaction between users and agents in a multiagent storytelling environment. In *Proceedings of the 2nd international joint conference on Autonomous agents and multiagent systems*, pages 741–748.

Riedl, M. O. and Bulitko, V. (2013). Interactive narrative: An intelligent systems approach. *Ai Magazine*, 34(1):67–67.

Riedl, M. O., Stern, A., Dini, D., and Alderman, J. (2008). Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*, 4(2):23–42.

Smith, D. E., Frank, J., and Cushing, W. (2008). The anml language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, volume 31.